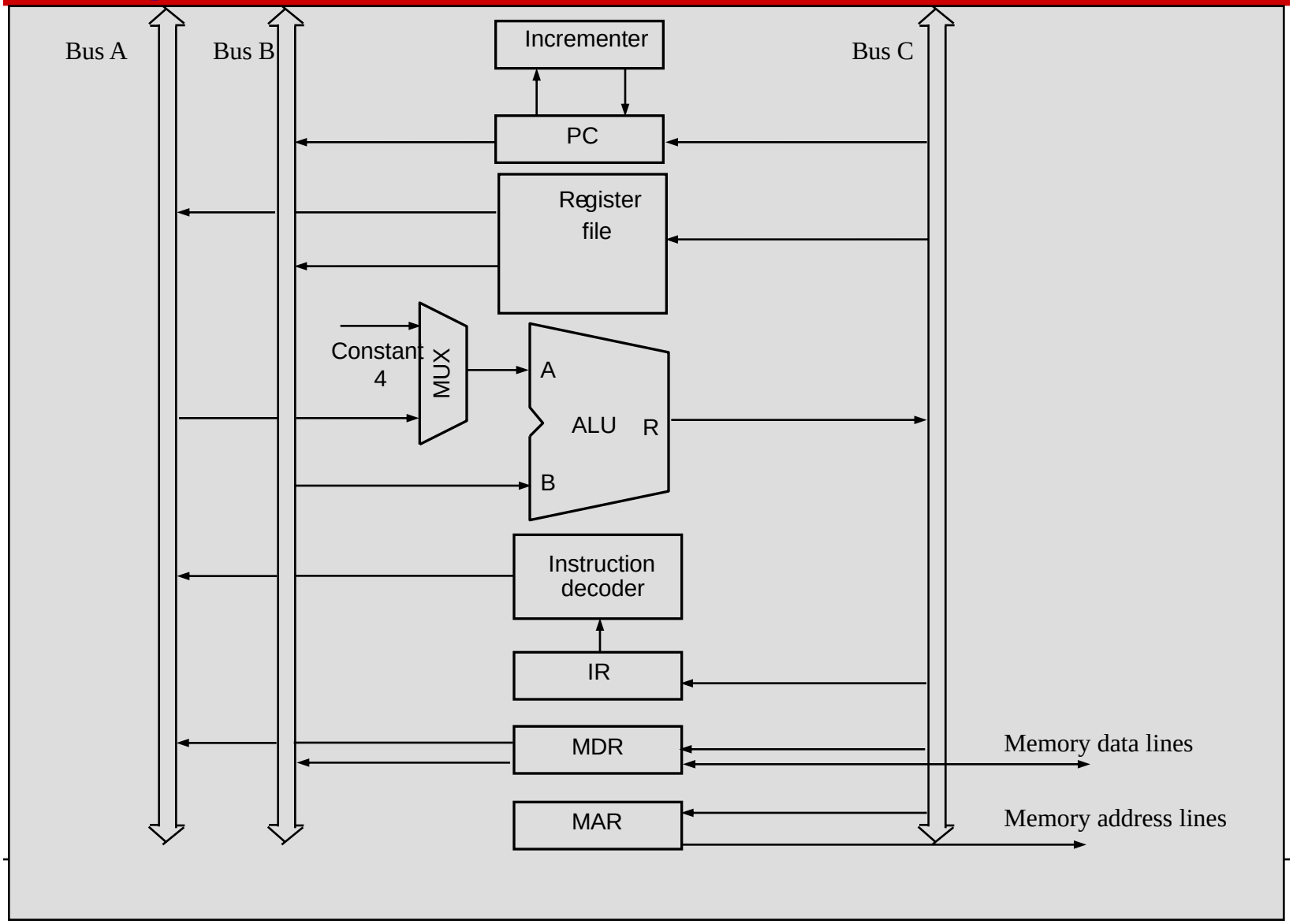
Lecture3(part2)

Topics covered:
CPU Architecture

Multiple-bus organization

- Simple single-bus structure:
  - Results in long control sequences, because only one data item can be transferred over the bus in a clock cycle.
- Most commercial processors provide multiple internal paths to enable several transfers to take place in parallel.
  - Multiple-bus organization.

Multiple bus organization (Three-bus organization)



Multiple bus organization (contd..)

- Three-bus organization to connect the registers and the ALU of a processor.
- All general-purpose registers are combined into a single block called register file.
  - Register file has three ports.
  - Two outputs ports connected to buses A and B, allowing the contents of two different registers to be accessed simultaneously, and placed on buses A and B.
  - Third input port allows the data on bus C to be loaded into a third register during the same clock cycle.
- Inputs to the ALU and outputs from the ALU:
  - Buses A and B are used to transfer the source operands to the A and B inputs of the ALU.
  - Result is transferred to the destination over bus C.

Multiple bus organization (contd..)

- ALU can also pass one of its two input operands unmodified if needed:
  - Control signals for such an operation are $R=A$ or $R=B$.
- Three bus arrangement obviates the need for Registers Y and Z in the single bus organization.
- Incrementer unit:
  - Used to increment the PC by 4.
  - Source for the constant 4 at the ALU multiplexer can be used to increment other addresses such as the memory addresses in multiple load/store instructions.

Multiple bus organization (contd..)

Three operand instruction: *ADD R4, R5, R6*

Step	Action
1	PC_{out}, R=B, MAR_{in}, Read, IncPC
2	WMFC
3	MDR_{outB}, R=B, IR_{in}
4	$R4_{outA}$, $R5_{outB}$, SelectA, Add, $R6_{in}$, End

1. Pass the contents of the *PC* through ALU and load it into *MAR*. Increment *PC*.
2. Wait for *MFC*.
3. Load the data received into *MDR* and transfer to *IR through ALU*.
4. Execution of the instruction is the last step.

Control unit

- To execute instructions the processor must generate the necessary control signals in proper sequence.
- Hardwired control:
  - Control unit is designed as a finite state machine.
  - Inflexible but fast.
  - Appropriate for simpler machines (e.g. RISC machines)
- Microprogrammed control:
  - Control path is designed hierarchically using principles identical to the CPU design.
  - Flexible, but slow.
  - Appropriate for complex machines (e.g. CISC machines)

Hardwired control

Step	Action
1	PC_{out} , MAR_{in} , Read, Select4,Add, Z_{in}
2	Z_{out} , PC_{in} , Y_{in} , WMFC
3	MDR_{out} , IR_{in}
4	$R3_{out}$, MAR_{in} , Read
5	$R1_{out}$, Y_{in} , WMFC
6	MDR_{out} , SelectY, Add, Z_{in}
7	Z_{out} , $R1_{in}$, End

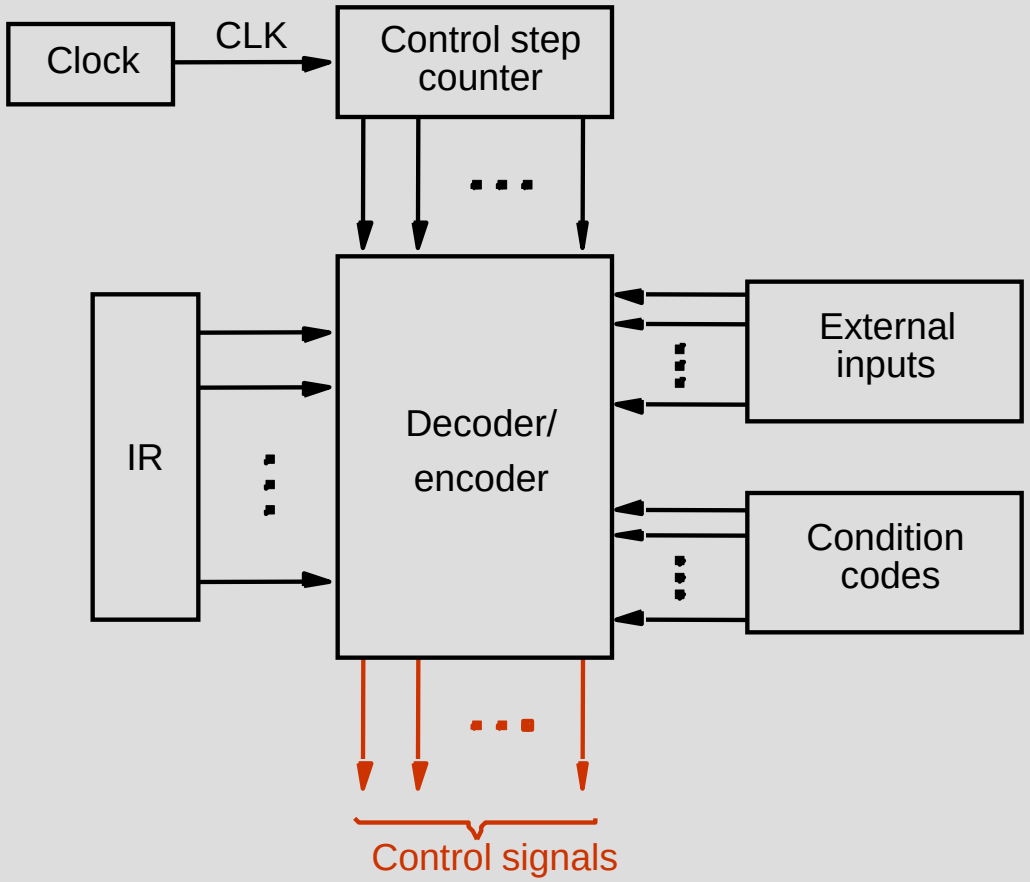
- Each step in this sequence is completed in one clock cycle.
- A counter may be used to keep track of the control steps.
- Each state or count, of this counter corresponds to one control step.

Hardwired control (contd..)

- Required control signals are determined by the following information:
  - Contents of the control step counter.
    - Determines which step in the sequence.
  - Contents of the instruction register.
    - Determines the actual instruction
  - Contents of the condition code flags.
    - Used for example in a BRANCH instruction.
  - External input signals such as MFC.

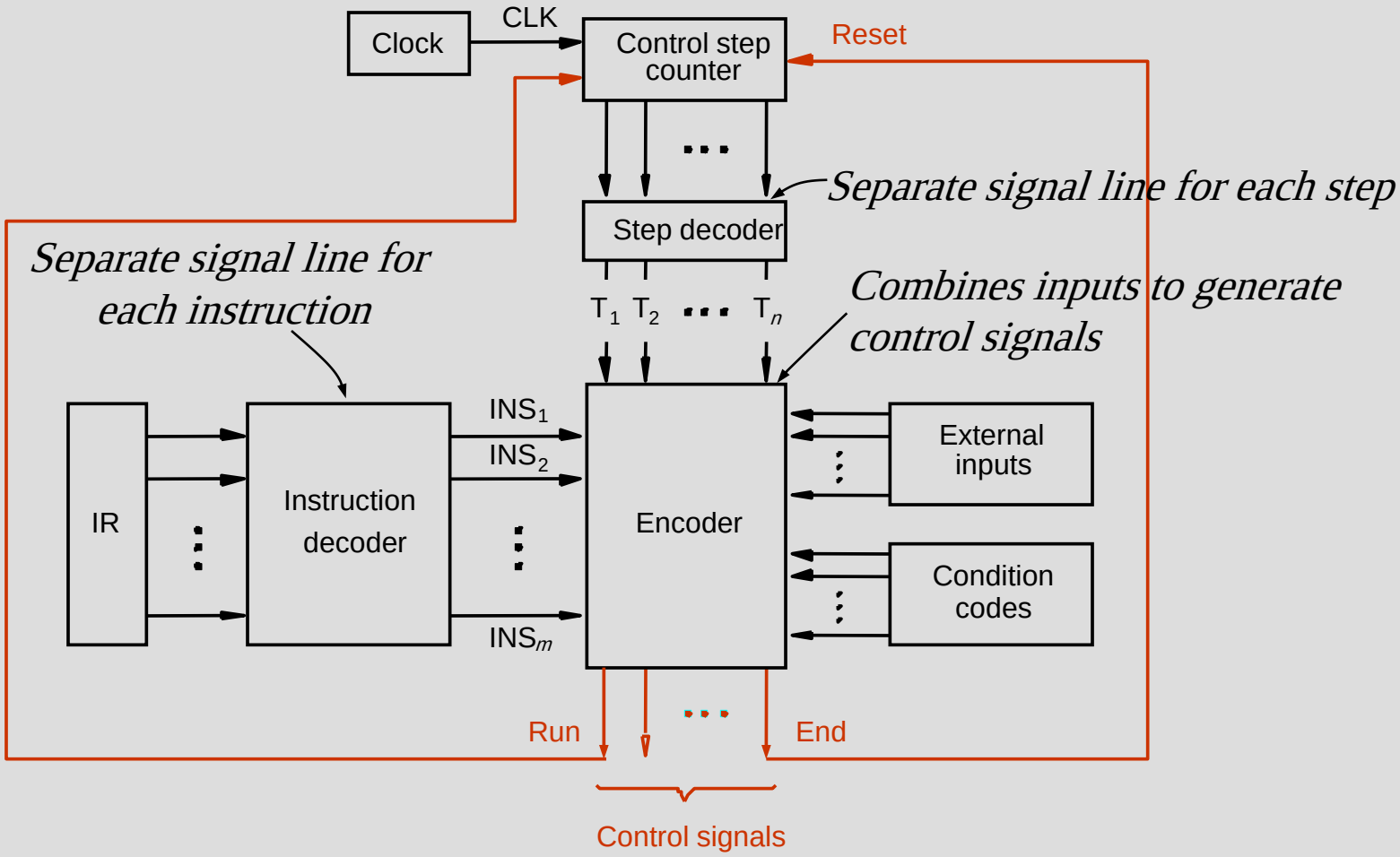
Hardwired control (contd..)

Control unit organization



- *Control unit consists of a decoder/encoder block to accept the following inputs:*
  - *Control step counter.*
  - *Instruction Register. IR*
  - *Condition codes*
  - *External inputs.*
- *Generates control signals.*

Hardwired control (contd..)

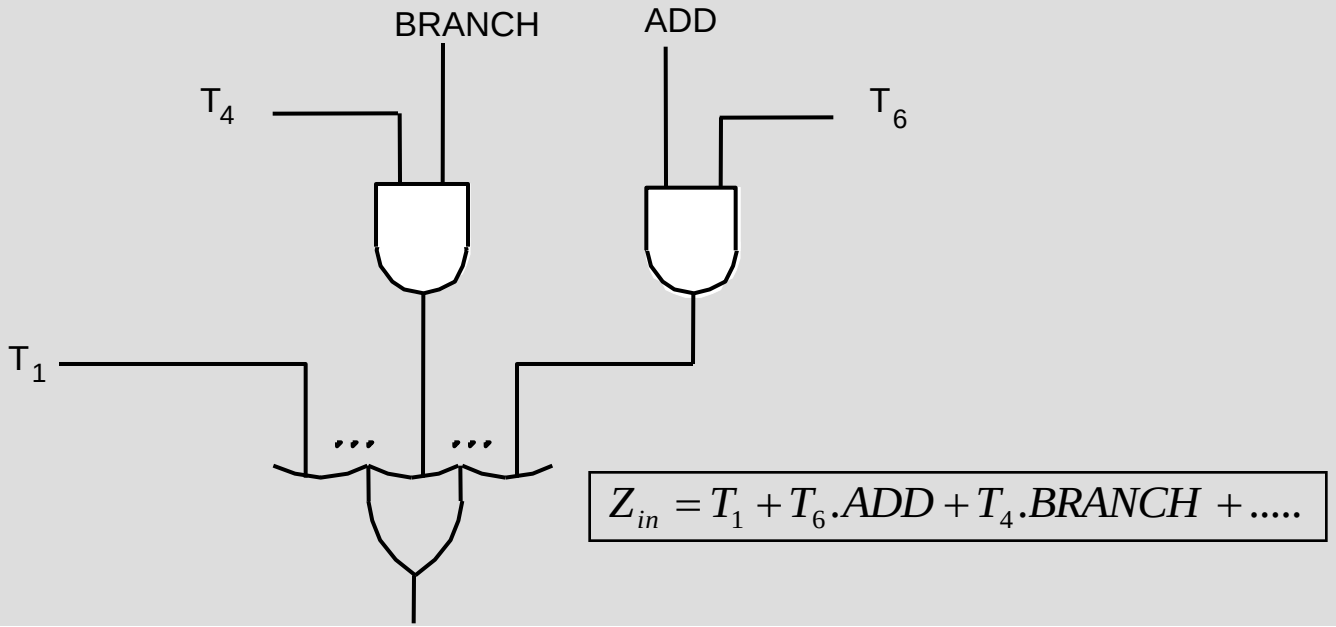


Hardwired control (contd..)

Control signals such as Z_{in}, PC_{out}, ADD are generated by encoder block

Suppose if Z_{in} is asserted:

- During T_1 for all instructions.
- During T_6 for ADD instruction.
- During T_4 for unconditional $BRANCH$ instruction
- .....



Hardwired control (contd..)

- Control hardware can be viewed as a state machine:
  - Changes state every clock cycle depending on the contents of the instruction register, condition codes, and external inputs.
- Outputs of the state machine are control signals:
- Sequence of control signals generated by the machine is determined by wiring of logic elements, hence the name "hardwired control".
- Speed of operation is one of the advantages of hardwired control is its speed of operation.
- Disadvantages include:
  - Little flexibility.
  - Limited complexity of the instruction set it can implement.

Microprogrammed control

- Hardwired control generates control signals using:
  - A control step counter.
  - Decoder/encode circuit.
- Microprogrammed control:
  - Control signals are generated by a ***program*** similar to machine language programs.

Microprogrammed control (contd..)

Control Word (**CW**) is a word whose individual **bits** represent various control signals.

Step	Action
1	PC_{out} , MAR_{in} , Read, Select4, Add, Z_{in}
2	Z_{out} , PC_{in} , Y_{in} , WMF C
3	MDR_{out} , IR_{in}
4	$R3_{out}$, MAR_{in} , Read
5	$R1_{out}$, Y_{in} , WMF C
6	MDR_{out} , SelectY, Add, Z_{in}
7	Z_{out} , $R1_{in}$, End

At every step, some control signals are asserted (=1) and all others are 0.

Control Signals:

- PC_{out}
- PC_{in}
- MAR_{in}
- ***Read***
- MDR_{out}
- IR_{in}
- Y_{in}
- ***SelectY***
- ***Select4***
- ***Add***
- Z_{in}
- Z_{out}
- $R1_{out}$
- $R1_{in}$
- $R3_{out}$
- ***WMFC***
- ***End .....***

Microprogrammed control (contd..)

Micro - instruction	..	PC_{in}	PC_{out}	MAR_{in}	Read	MDR_{out}	IR_{in}	Y_{in}	Select	Add	Z_{in}	Z_{out}	$R1_{out}$	$R1_{in}$	$R3_{out}$	WMFC	End	..
1		0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	
2		1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	
3		0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	
4		0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	
5		0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	
6		0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	
7		0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	

- *At every **step**, a Control Word needs to be generated.*
- *Every **instruction** will need a sequence of CWs for its execution.*
- *Sequence of **CWs** for an **instruction** is the **microroutine** for the instruction.*
- *Each **CW** in this microroutine is referred to as a **microinstruction**.*

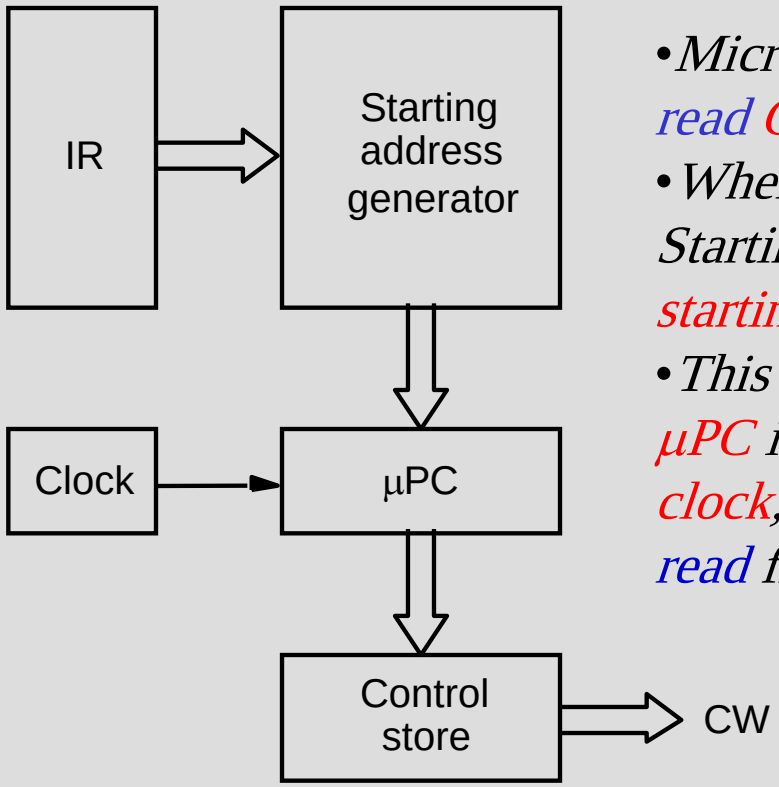
(SelectY is represented by Select=0, & Select4 by Select=1)

Microprogrammed control (contd..)

- Every instruction will have its own microroutine which is made up of microinstructions.
- Microroutines for **all** instructions in the instruction set of a computer are stored in a special memory called Control Store.
- Recall that the Control Unit generates the control signals:
  - Sequentially reading the CWs of the corresponding microroutine from the control store.

Microprogrammed control (contd..)

Basic organization of a microprogrammed control unit.



- *Microprogram counter (µPC) is used to read CWs from control store sequentially.*
- *When a new instruction is loaded into IR, Starting address generator generates the starting address of the microroutine.*
- *This address is loaded into the µPC. µPC is automatically incremented by the clock, so successive microinstructions are read from the control store.*

Microprogrammed control (contd..)

- Basic organization of the microprogrammed control unit cannot check the status of condition codes or external inputs to determine what should be the next microinstruction.
- Recall that in the hardwired control, this was handled by an appropriate logic function.
- How to handle this in microprogrammed control:
  - Use conditional branch microinstructions.
  - These microinstructions, in addition to the branch address also specify which of the external inputs, condition codes or possibly registers should be checked as a condition for branching.

Microroutine for the instruction (Branch < 0)

Address	Microinstruction
0	PC_{out} , MAR_{in} , Read, Select4, Add, Z_{in}
1	Z_{out} , PC_{in} , Y_{in} , WMFC
2	MDR_{out} , IR_{in}
3	Branch to starting address of appropriate microroutine
. . . . .	
25	If N=0, then branch to microinstruction 0
26	Offset-field-of-IR_{out} , Select Y, Add, Z_{in}
27	Z_{out} , PC_{in} , End

Fetch *BRANCH<0* instruction, microroutine is at address 25.

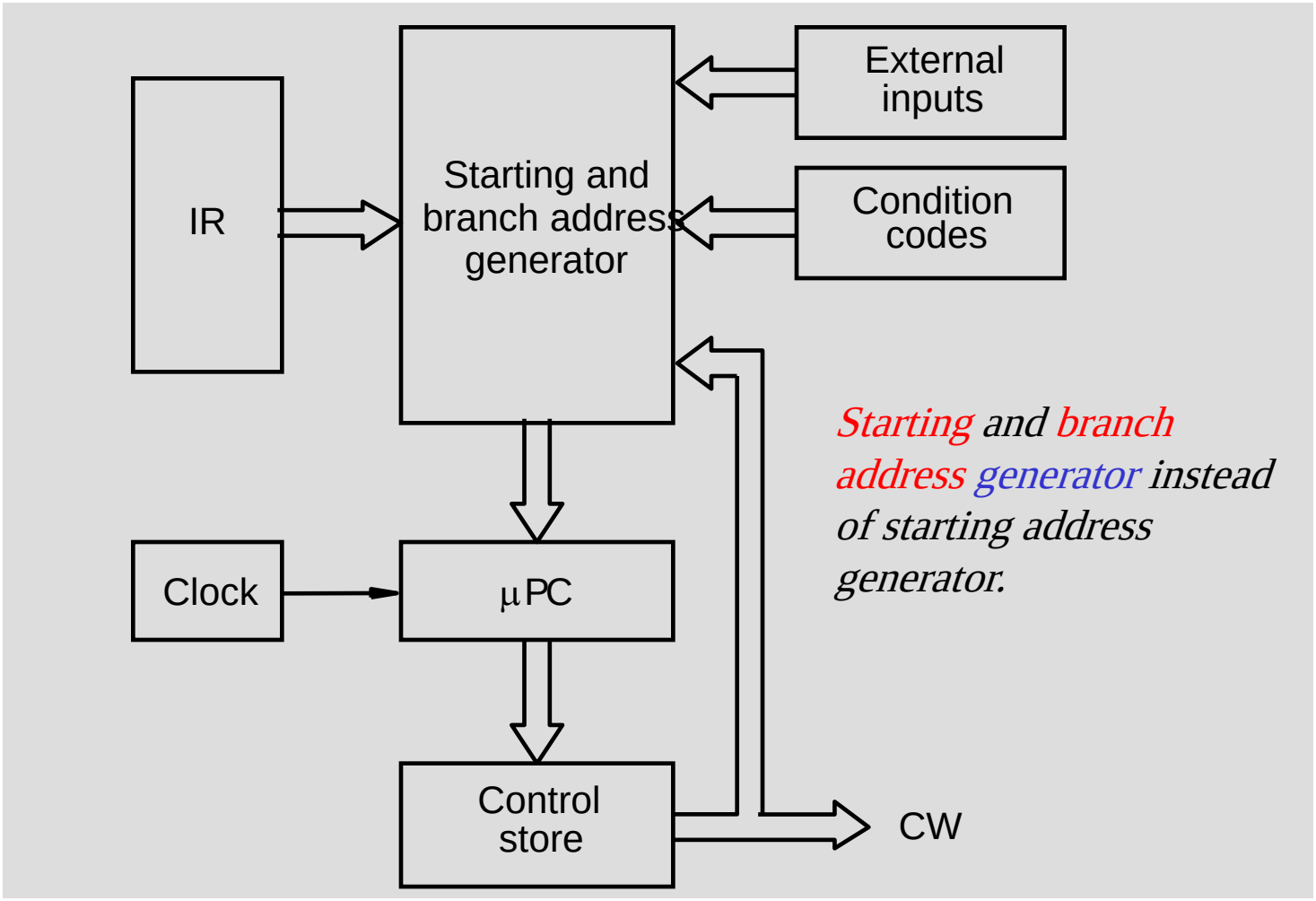
Branch to address 25.

Test the *N* bit of the condition codes
If 0, go to 0 and get new instr.
Else execute microinstruction located at 26 and put the branch target address into Register *Z*.
*(*Microinstruction at location 27*)*.

Address 25 is the output of starting address generator and is loaded into the microprogram counter (*μPC*).

Microprogrammed control (contd..)

Control Unit with Conditional Branching in the Microprogram



Microprogrammed control (contd..)

- Starting and branch address generator accepts as inputs:
  - Contents of the Instruction Register **IR** (as before).
  - External inputs
  - Condition codes
- Generates a new address and loads it into microprogram counter (μPC) when a microinstruction instructs it do so.
- μPC is incremented every time a microinstruction is fetched except:
  - New instruction is loaded into IR, μPC is loaded with the starting address of the microroutine for that instruction.
  - Branch instruction is encountered and branch condition is satisfied, μPC is loaded with the branch address.
  - End instruction is encountered, μPC is loaded with the address of the first CW in the microroutine for the instruction fetch cycle.

Microprogrammed control (contd..)

Microinstruction format

- Simple approach is to allocate one bit for each control signal
  - Results in long microinstructions, since the number of control signals is usually very large.
  - Few bits are set to 1 in any microinstruction, resulting in a poor use of bit space.

- Reduce the length of the microinstruction by taking advantage of the fact that most signals are not needed simultaneously, and many signals are mutually exclusive.

- For example:
  - Only one ALU function is active at a time.
  - Source for a data transfer must be unique.
  - *Read* and *Write* memory signals cannot be active simultaneously.

Microinstruction format

- Group mutually exclusive signals in the same group.
- At most one microoperation can be specified per group.
- Use binary coding scheme to represent signals within a group.

Examples:

- If ALU has 16 operations, then 4 bits can be sufficient.
- Group register output signals into the same group, since only one of these signals will be active at any given time (Why?)

If the CPU has 4 general purpose registers, then PC_{out}, MDR_{out}, Z_{out}, $Offset_{out}$, $R0_{out}$, $R1_{out}$, $R2_{out}$, $R3_{out}$ and $Temp_{out}$ can be placed in a single group, and 4 bits will be needed to represent these.

Microprogrammed control (contd..)

Microinstruction

F1	F2	F3	F4	F5	

F1 (4 bits)	F2 (3 bits)	F3 (3 bits)	F4 (4 bits)	F5 (2 bits)
0000: No transfer	000: No transfer	000: No transfer	0000: Add	00: No action
0001: PC_{out}	001: PC_{in}	001: MAR_{in}	0001: Sub	01: Read
0010: MDR_{out}	010: IR_{in}	010: MDR_{in}	⋮	10: Write
0011: Z_{out}	011: Z_{in}	011: $TEMP_{in}$	1111: XOR	
0100: $R0_{out}$	100: $R0_{in}$	100: Y_{in}	16 ALU functions	
0101: $R1_{out}$	101: $R1_{in}$			
0110: $R2_{out}$	110: $R2_{in}$			
0111: $R3_{out}$	111: $R3_{in}$			
1010: $TEMP_{out}$				
1011: $Offset_{out}$				

F6	F7	F8	...

F6 (1 bit)	F7 (1 bit)	F8 (1 bit)
0: SelectY	0: No action	0: Continue
1: Select4	1: WMFC	1: End

- *Each group occupies a large enough field to represent all the signals.*
- *Most fields must include one inactive code, which specifies no action.*
- *All fields do not have to include inactive code.*

Vertical and Horizontal Organization

In vertical organization :

- Each microinstruction contains a small number of control functions leading to more microinstructions required for the execution of each instruction.
- results in slower operating speeds.

- In horizontal organization :
- Many resources can be controlled with a single instruction leading into small number of microinstructions required for the execution of each instruction.
- require that the CPU structure allows parallel use of resources.
- results in higher operating speed